CPP05 – Ex00 : Bureaucrat

Fiche explicative pédagogique + démonstrations concrètes

1. Objectif pédagogique de l'exercice

Cet exercice introduit les fondations de la Programmation Orientée Objet en C++ : création de classes, gestion d'erreurs avec exceptions, forme canonique et surcharge d'opérateur.
L'objectif est d'apprendre à structurer son code de manière professionnelle et sécurisée.

2. Une classe = un modèle d'objet

La classe **Bureaucrat** décrit un personnage : son nom et son grade. Une fois créée, elle permet d'instancier autant de bureaucrates que nécessaire.

Exemple concret :

```
Bureaucrat a("Alice", 42); std::cout << a;
```

Résultat attendu :

```
Alice, bureaucrat grade 42
```

Contre-exemple (grade impossible) :

```
Bureaucrat a("Alice", -5);
```

→ Erreur : *Bureaucrat grade too high*

3. Comprendre la limite des grades

Le grade est toujours entre **1** (meilleur grade) et **150** (pire grade). Cette contrainte garantit l'intégrité de l'objet.

Exemple valide :

```
Bureaucrat g("John", 100);
```

Exemple invalide :

```
Bureaucrat b("John", 151);
```

→ Erreur : *Bureaucrat grade too low*

4. Monter et descendre en grade

Modifier le grade signifie vérifier que la nouvelle valeur reste dans la plage autorisée.

Montée autorisée :

```
Bureaucrat b("Bob", 2); b.incrementGrade();
```

→ Bob passe au grade 1

Montée interdite :

```
Bureaucrat b("Bob", 1); b.incrementGrade();
```

→ Exception levée : grade too high

Descente autorisée :

```
Bureaucrat c("Cara", 149); c.decrementGrade();
```

→ Cara passe au grade 150

Descente interdite :

```
Bureaucrat c("Cara", 150); c.decrementGrade();
```

→ Exception levée : grade too low

5. Les exceptions : sécuriser le programme

Les exceptions permettent d'empêcher un programme de continuer dans un état invalide. Elles signalent clairement l'erreur et permettent au développeur d'agir en conséquence.

Exemple avec try/catch :

```
try { Bureaucrat x("Xavier", 0); } catch (std::exception &e;) { std::cout
<< "Erreur : " << e.what() << std::endl; }
```

Sans try/catch → crash garanti.

6. Forme canonique : bien gérer la copie

La forme canonique impose 4 fonctions : constructeur par défaut, constructeur de copie, opérateur d'affectation, destructeur. Elle garantit un comportement correct lors des copies.

Exemple de copie valide :

```
Bureaucrat a("Anna", 10); Bureaucrat b(a);
```

7. Surcharge de l'opérateur <<

Cela permet d'afficher un bureaucrate avec la syntaxe naturelle :

```
std::cout << b;
```

Erreur courante :

```
Bureaucrat* b = new Bureaucrat("Dylan", 42); std::cout << b; // affiche
une adresse !
```

Solution : utiliser *b

Résumé final

Cet exercice enseigne comment créer une classe solide, protéger ses données internes, empêcher les états invalides, gérer les erreurs proprement et structurer un code de type professionnel en C++98.